



Output Subsystem Status and Plans

B. W. Bush

Los Alamos National Laboratory

19 March 1997



Abstract

The output subsystem collects data from a running microsimulation, stores the data for future use, and manages the subsequent retrieval of the data. We discuss the capabilities and uses of the current simulation output subsystem and the enhancements presently underway and planned.



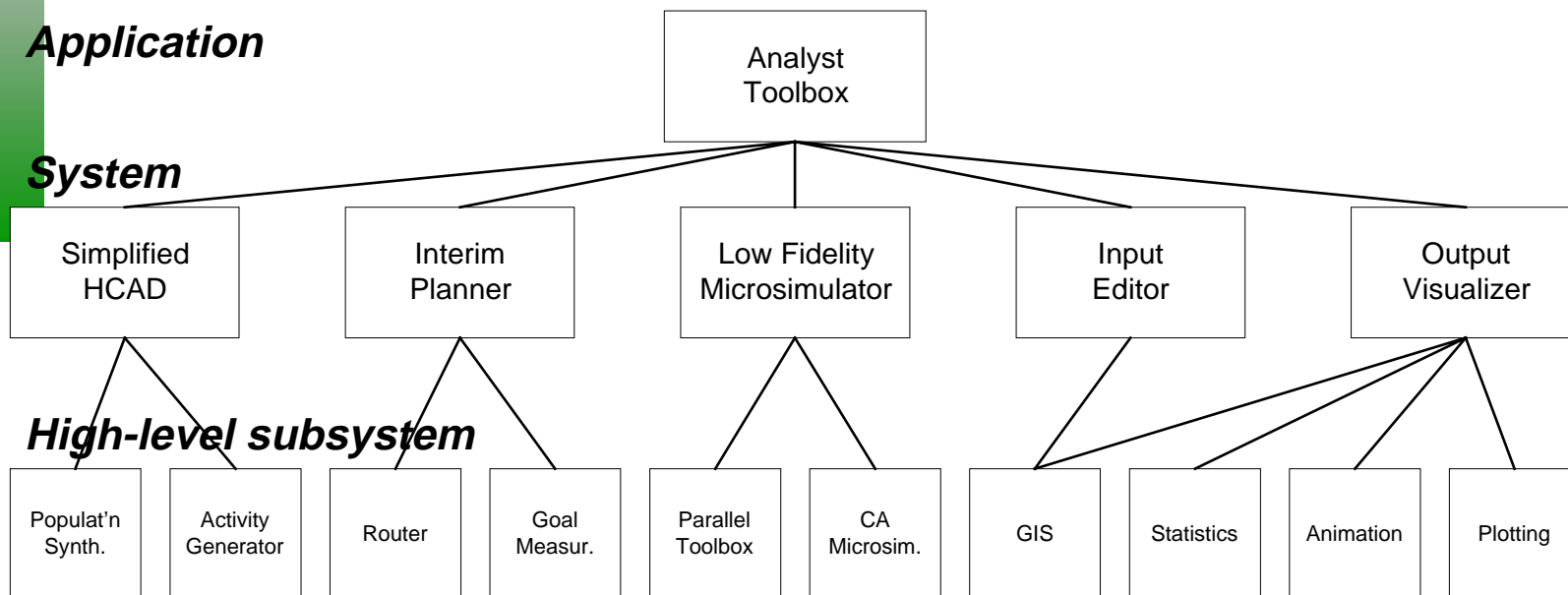
Outline

- *current status*
 - *capabilities*
 - *uses*
- *work underway (short term)*
 - *new data collection*
 - *performance enhancements*
- *planned research (long term)*
 - *real-time analysis*
 - *feature recognition*

Simulation Output Subsystem

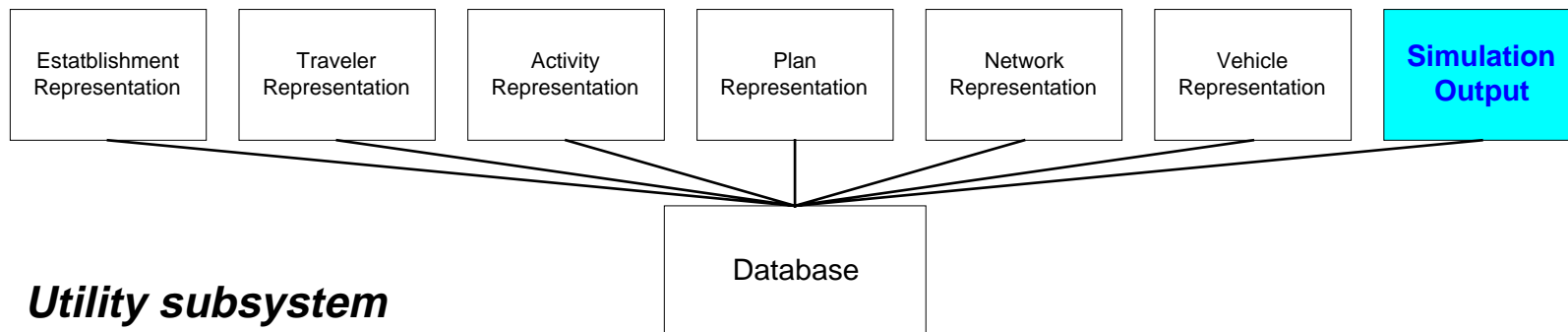
Application

System



High-level subsystem

Low-level subsystem

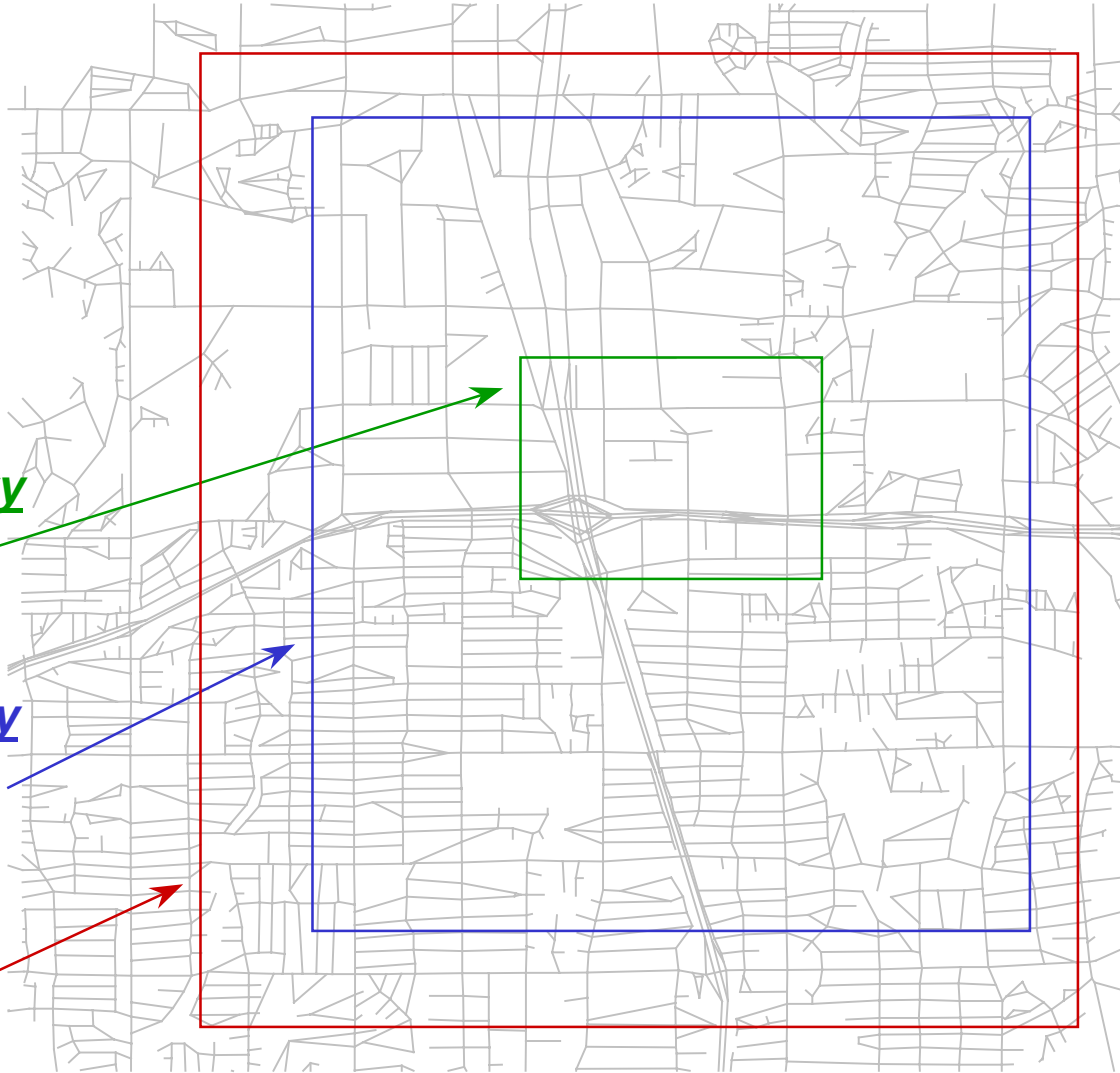


Utility subsystem

Current Capabilities

- *user configurable*
 - *choice of what to collect*
 - *possible filtering by space (nodes and links) and by time*
- *several types of data collected*
 - *trajectory/evolution data (collected for each time step)*
 - *vehicles on links (location, velocity, and status)*
 - *vehicles in intersections (location)*
 - *traffic controls (phase and allowed movements)*
 - *event data (collected when event occurs)*
 - *vehicle status (lost vehicle, off plan, etc.)*
 - *summary data (sampled and reported periodically)*
 - *link travel times (count, mean, and variance of traversal time)*
 - *link densities (count and mean velocities in ℓ -meter “boxes”)*
- *distributed data collection (no computer network overhead)*
- *general capability (not tied to specific type of microsimulation)*

User Configurability

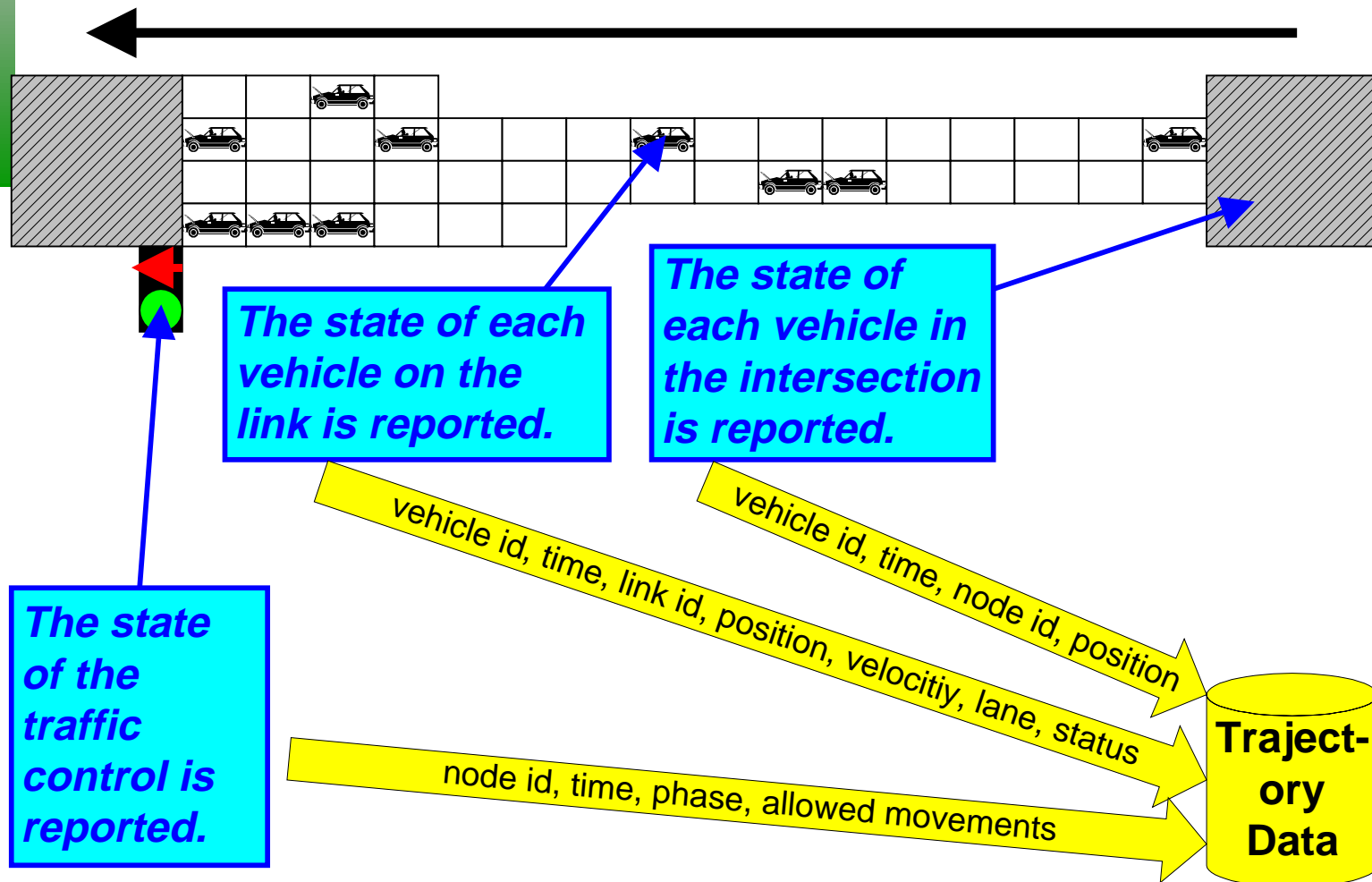


collect trajectory
data here from
8 AM to 9 AM

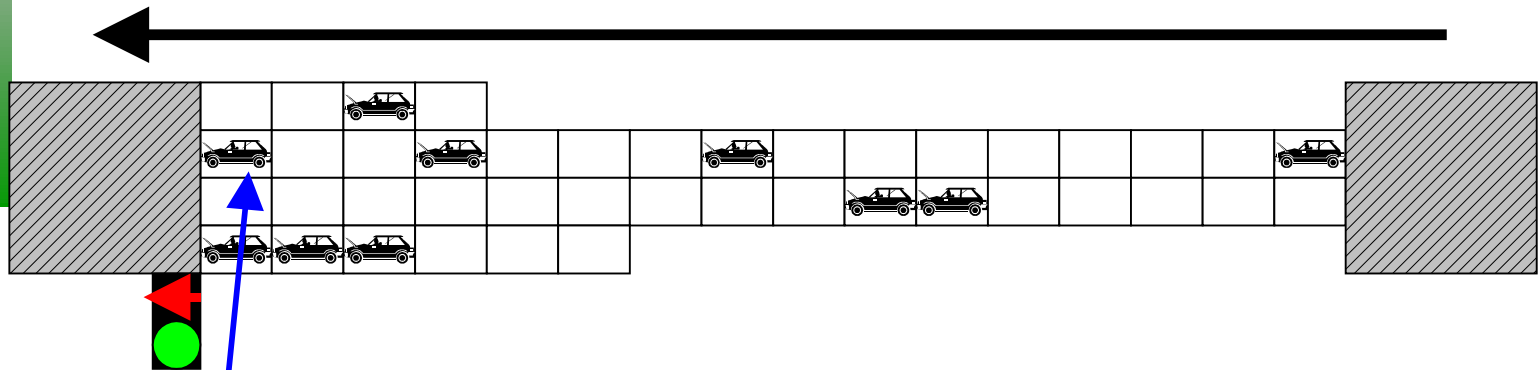
collect summary
data here from
6 AM to 10 AM

collect event
data here at
all times

Trajectory/Evolution Data



Event Data

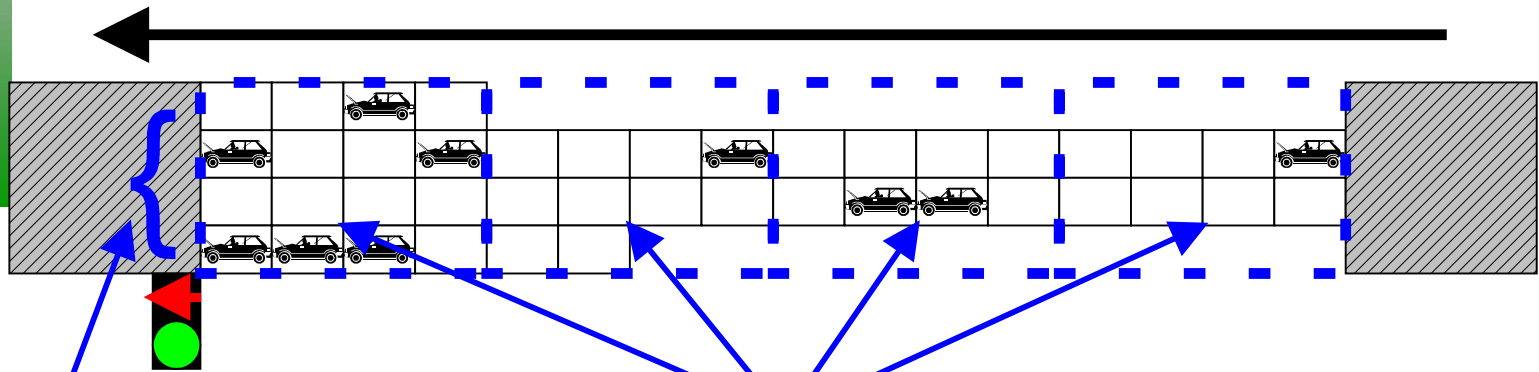


The vehicle has just become lost because it cannot make the left turn it planned on making at this intersection. This event is reported.

vehicle id, time, link id, position, velocity, lane, status

Event Data

Summary Data

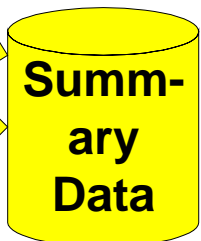


The traversal times for vehicles that have traveled the length of the link are summarized.

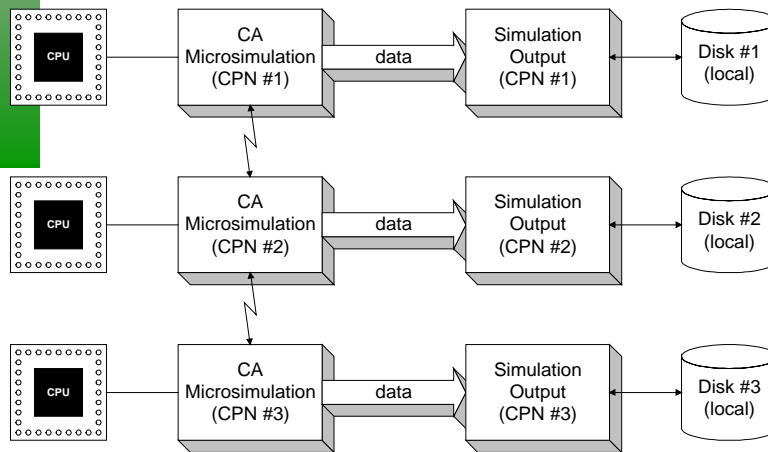
The vehicle counts and velocities in "boxes" along the link are summarized.

link id, box position, vehicle count, sum of velocities

link id, vehicle count, sum of travel times

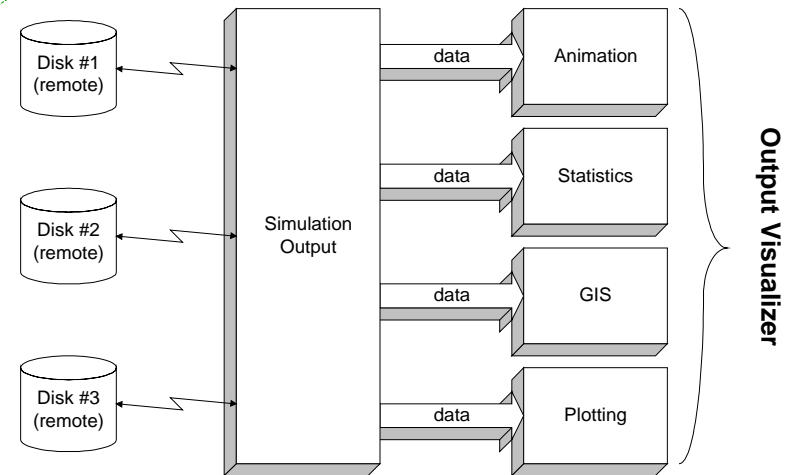


Data Collection and Retrieval



collection

retrieval





Current Uses

- *trajectory/evolution data*
 - *animating vehicle movement*
 - *making “snapshots” of the traffic periodically*
 - *understanding the traffic behavior induced by CA rules*
 - *refining the driving logic*
 - *deriving fundamental diagrams*
 - *measuring turn counts*
- *event data*
 - *locating problems with driver logic*
 - *locating problems with plans*
- *summary data*
 - *animating vehicle densities*
 - *identifying congestion and deadlocks*
 - *replanning trips using observed link travel times*

New Data Collection

■ *summary data*

- *turn counts*
 - *tally of vehicles making movements at intersections*
 - *customized reporting periods*
- *velocity-acceleration matrices (input for air quality calculations)*
 - *tally of vehicle velocity versus acceleration*
 - *customized reporting periods and spatial bins*

NEW!

■ *cell data*

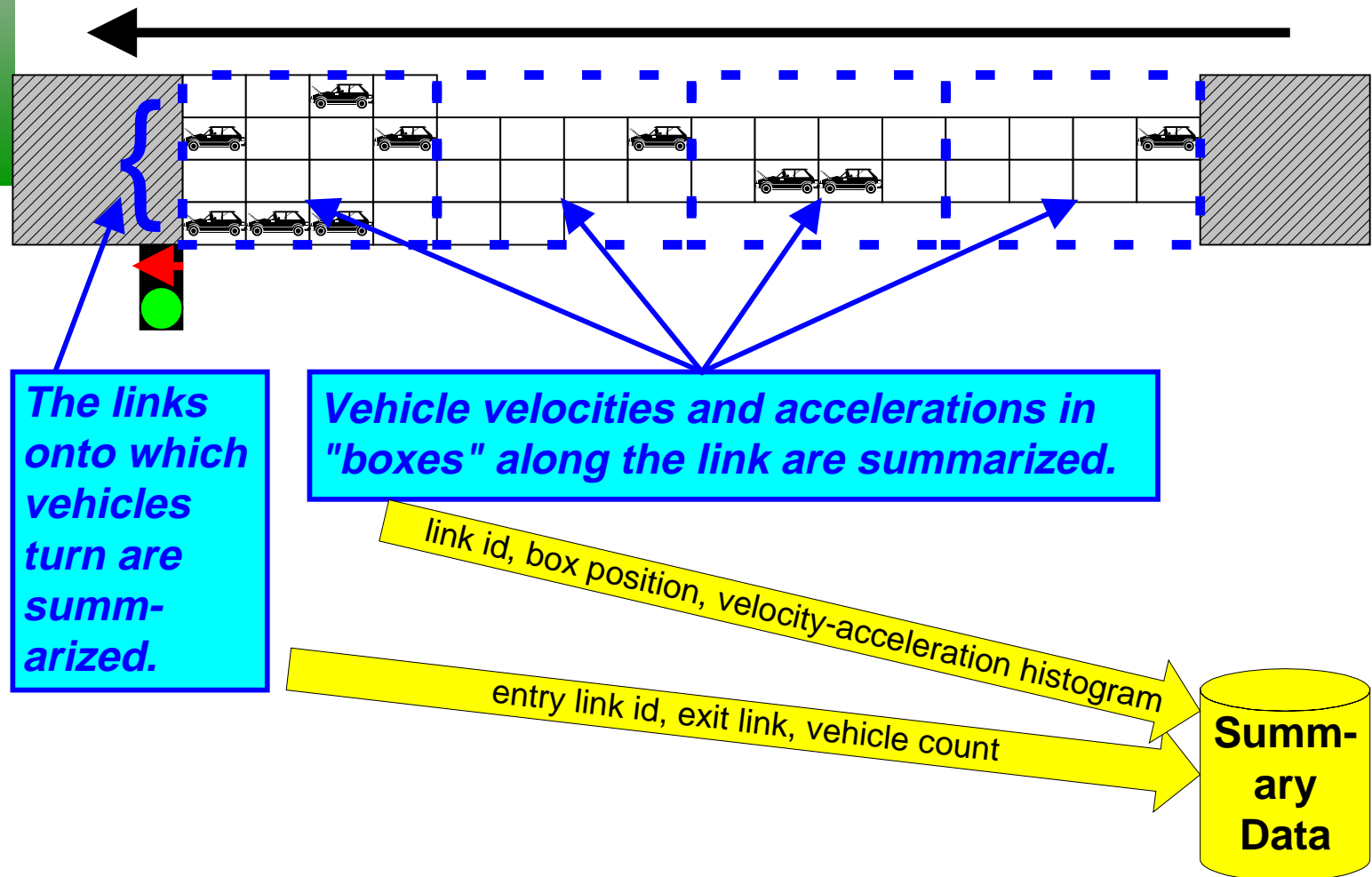
- *fundamental diagrams*
 - *flow, density, and velocity (mean and variance)*
 - *customized reporting periods and spatial location*

NEW!

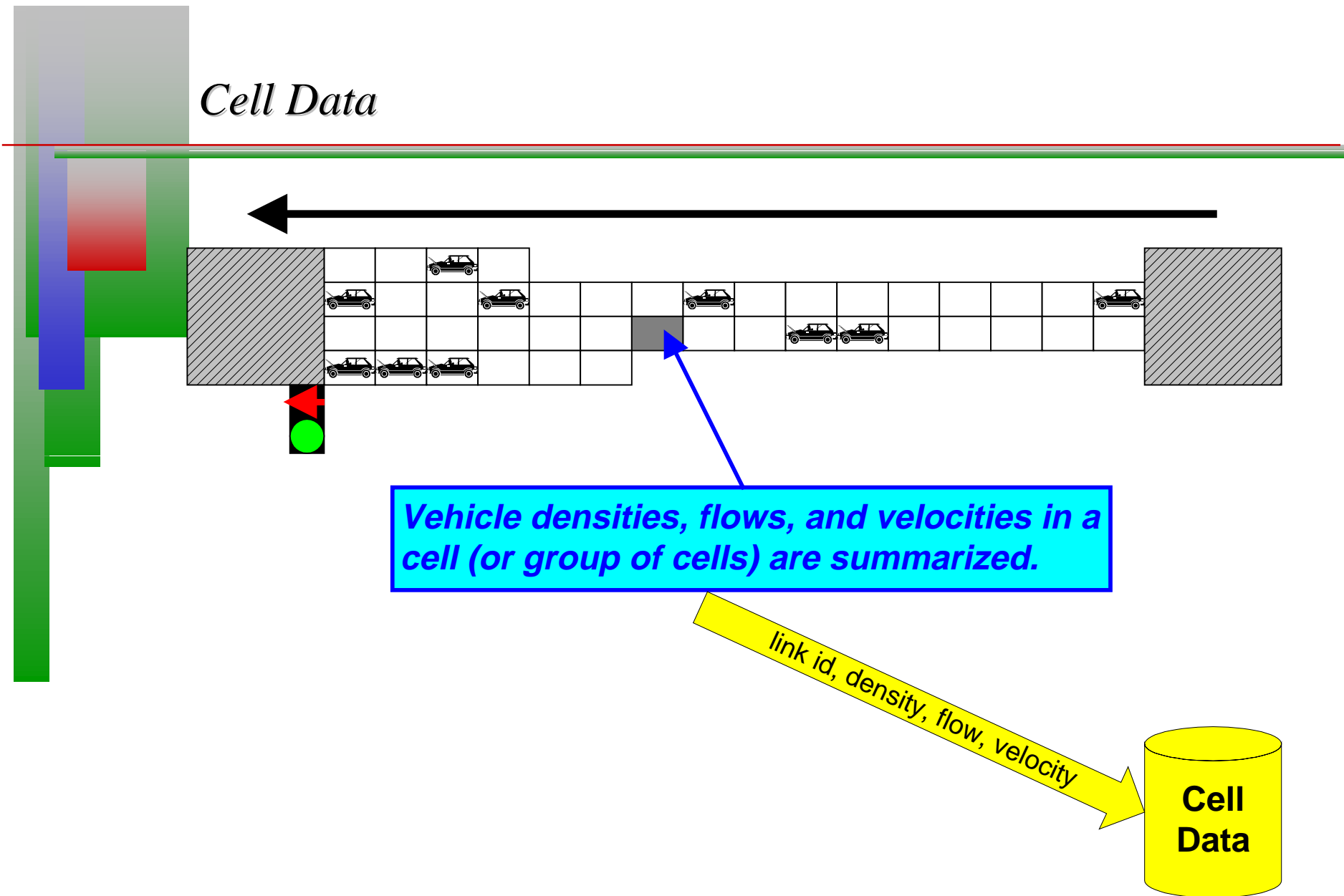
■ *grid data*

- *vehicle animation*
 - *location and speeds of vehicles as a function of time*
 - *high-performance compressed/indexed format*

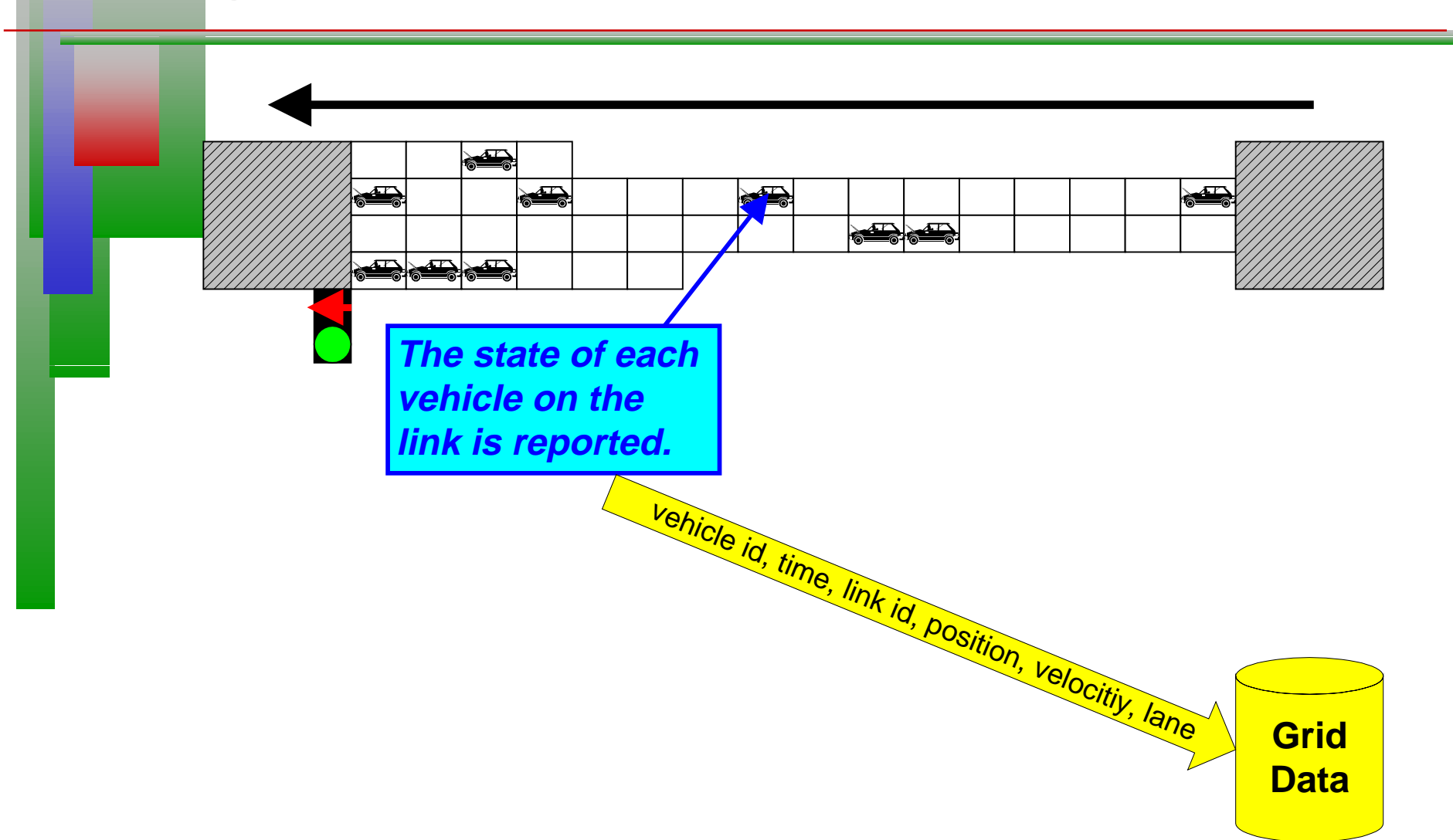
New Types of Summary Data



Cell Data



Grid Data





Performance Enhancements

- *compression*
 - *speeds simulation and retrieval by reducing I/O wait times*
 - *shrinks size of output files on disk*
- *sorting*
 - *organizes data efficiently as it is retrieved*
 - *simplifies post-retrieval analysis process*
- *filtering*
 - *eliminates unwanted data during retrieval*
 - *simplifies post-retrieval analysis process*
- *indexing*
 - *speeds sorting and filtering*
 - *allows for “random access” searches*

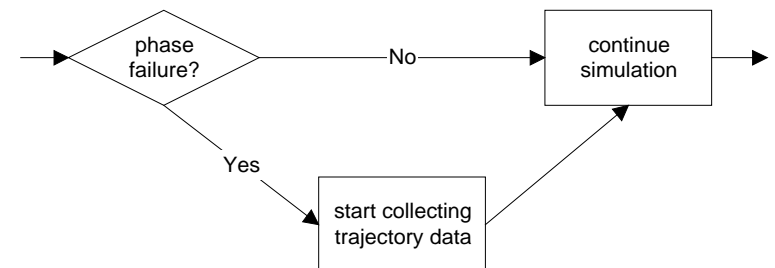
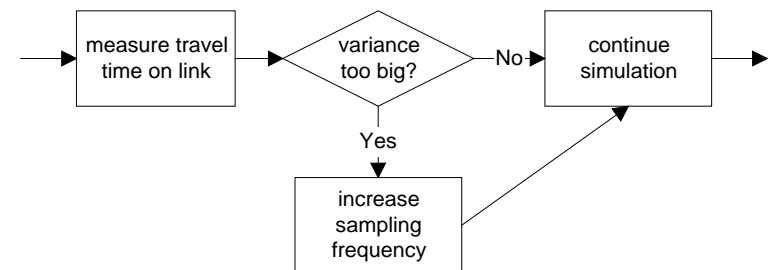
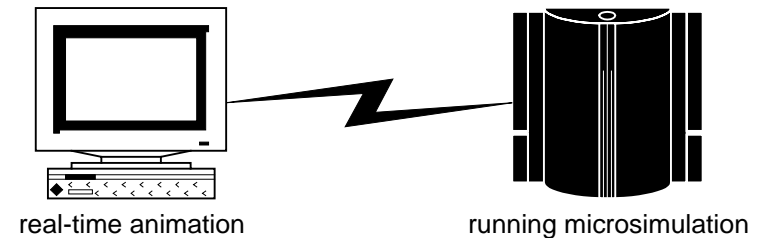


Real-Time Analysis

- ***real-time*** or ***on-line*** analysis
 - *make data available while the simulation is still running*
 - *adapt the data collection to the conditions in the simulation*
 - *analyze and view the data as it is generated*
- ***adaptive*** data collection
 - *let certain events act as **triggers** that change how data collection proceeds*
 - *recognize **patterns** and **features** in the data as they form and report them to the user*

Examples of Real-Time Analysis

- *users might view animation of the simulation while it runs*
- *the sampling frequency for data might be varied automatically to reduce the variance in statistical estimators*
- *the occurrence of a phase failure at a traffic light might trigger more extensive data collection at that location for the remainder of the simulation*





Feature Recognition

- *automatic recognition of traffic patterns and features*
 - *trigger adaptive data collection*
 - *assist “debugging” network, plans, and driving logic*
 - *aid understanding and analysis of simulation output*
- *example patterns/features of interest*
 - *jammed link*
 - *jammed intersection*
 - *gridlock*
 - *shock wave*
 - *vehicle clustering*
 - *phase failure at light*
 - *shear zone*
 - *lane change zone*



Current Status

- *The simulation output subsystem has supported the data collection needs of the interim operational capability and the case study.*
- *Based on needs identified during the case study, we are presently implementing new types of data collection and enhancing the performance of the subsystem to make it easier for analysts to work with simulation output and get the data they need.*
- *We have plans to develop real-time and adaptive output collection capabilities.*
- *Work on the simulation output subsystem will continue throughout the TRANSIMS development process.*



Data Sampling and Storage

B. W. Bush

Los Alamos National Laboratory

19 March 1997



Abstract

This presentation outlines an integrated plan of research and development for planner and simulation output data sampling and storage in TRANSIMS. The approach focuses on the flow of information in simulations and on the extraction of features from output data. It will improve our understanding of how to extract traffic features (such as jams, incidents, flows, phase failures, etc.), to optimize traffic sensor placement, and to store planner and simulation output data efficiently.



Outline

- *goals*
- *motivation*
- *approach*
- *applications*
 - *compressed storage of planner output*
 - *information content of microsimulation output*
 - *scaling behavior in microsimulation output*
- *status*



Goals

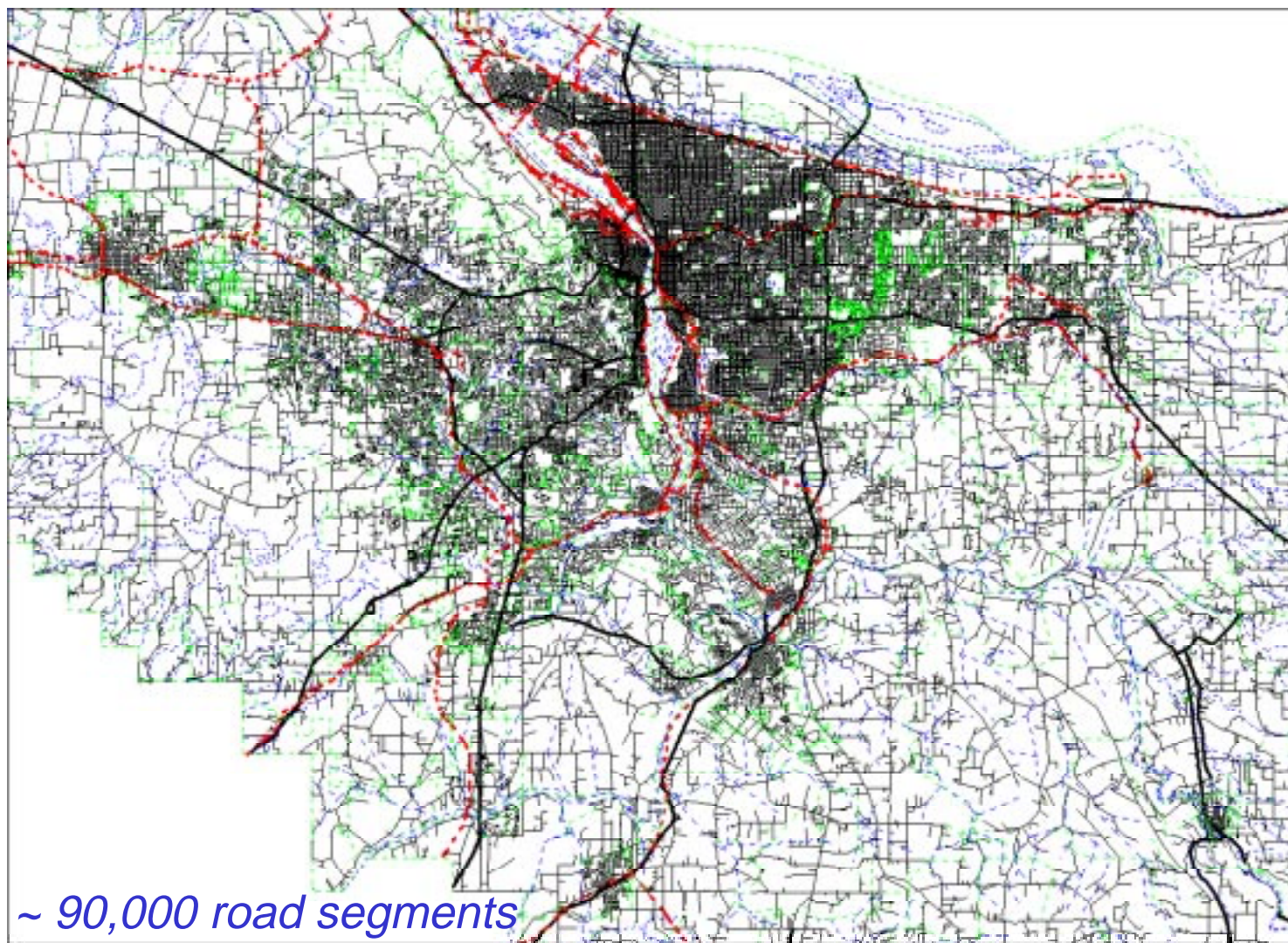
- *To understand how information flows in simulations in order to deal with the simulation data sampling and storage problem.*
- *To develop an abstract methodology for . . .*
 - *evaluating sampling techniques,*
 - *estimating traffic observables, and*
 - *recognizing traffic features.*
- *To develop concrete methods to . . .*
 - *recognize patterns (traffic features) in the microsimulation output,*
 - *collect specific new forms of microsimulation output, and*
 - *improve the performance of microsimulation output collection and storage.*
- *To reduce the size of the planner output files without adversely affecting the performance of the planner or the microsimulation.*



Data Volume

- *Approximately 30-60 bytes of data are needed to describe the state of a vehicle at any given time in the simulation.*
- *Data can conceivably be collected for each vehicle every time it is moved (currently, once per second).*
- *In the largest metropolitan areas, it is possible for 1,000,000 vehicles to be moving on the road network simultaneously.*
- *This means that a four-hour simulation could require 400-800 GB of storage if all trajectory/evolution data is stored.*
- *Conclusions:*
 - *We have to store efficiently large amounts of data.*
 - *It also must have the capability to not store data when it is not necessary to do so.*

Portland Area



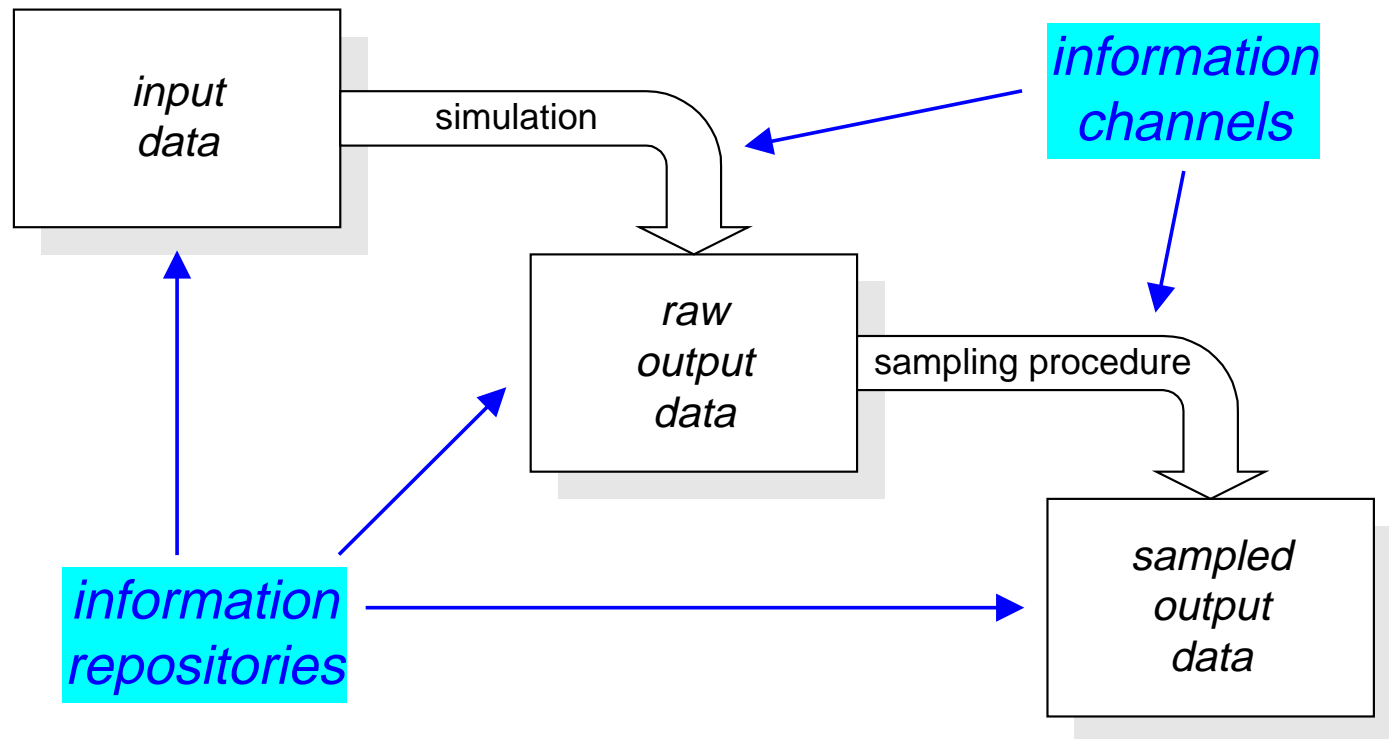


Motivation

- *The amount of data generated by a simulation is enormous.*
- *Even if one could store all of the generated data, it would be difficult to organize, analyze, and visualize so much data.*
- *We need procedures to sample the simulation data.*
- *It is necessary to develop a methodology for comparing and evaluating different sampling procedures.*
- *A key aspect of this measurement problem is understanding how much information is actually collected by a given sampling procedure.*

Sampling

- An **information sampling procedure** is simply an algorithm for making measurements from output data.
- Processes such as simulation or sampling are channels through which information flows.





Basic Approach

- *Develop a means to trace the flow of information in simulations.*
- *Construct a methodology for evaluating sampling procedures and comparing them quantitatively.*
- *Investigate optimization techniques for generating sampling schemes.*
- *Apply this methodology to specific sampling and storage problems in TRANSIMS.*



Some Fields with Similar Data Volume Problems

■ *experiments*

- *high-energy physics*

■ *computation*

- *logistics simulation*
- *computational fluid dynamics*
- *molecular dynamics computations*
- *lattice Boltzmann calculations*

■ *information science*

- *data mining*
- *pattern recognition*
- *classification*



Some Analysis Techniques

- *information theory*
 - *coding techniques*
 - *information measures*
 - *mutual information*
- *statistics*
 - *Bayes classifiers*
 - *polynomial regression*
 - *radial basis functions*
 - *hypothesis testing*
 - *nearly sufficient statistics*
- *systems theory*
 - *symbol dynamics*
 - *multi-layer perceptrons*
 - *adaptive algorithms*

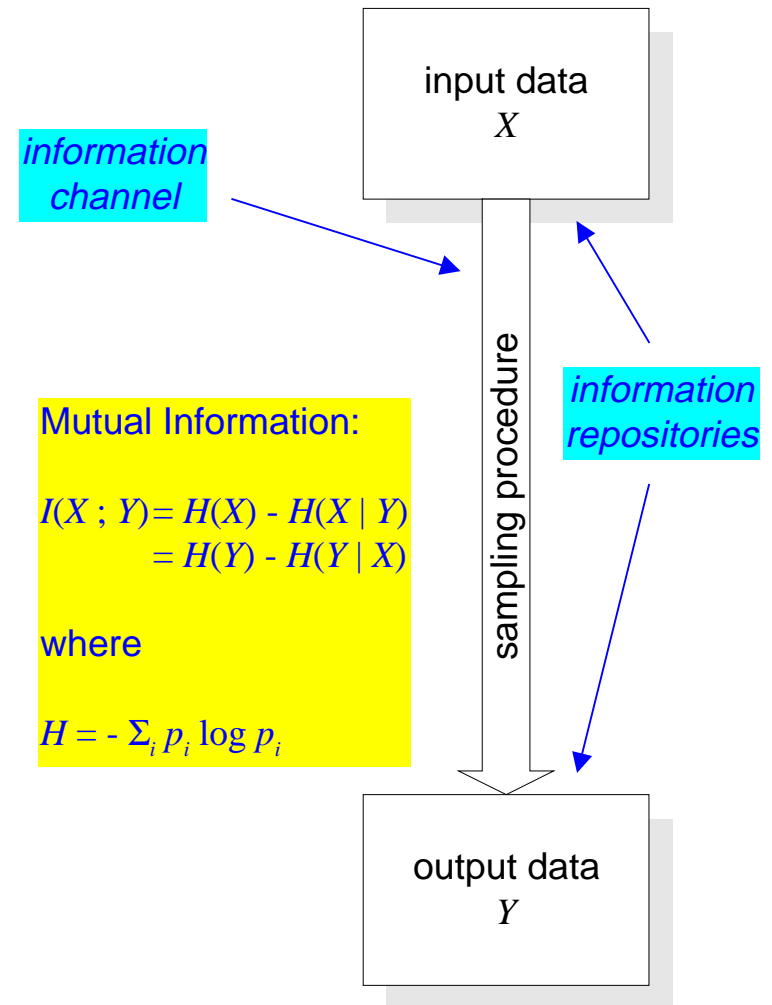


Information-Theoretic Approach

- *Treat the simulation as an information source.*
 - *Develop means for measuring its information content.*
- *Treat the sampling procedure as a communication channel.*
 - *Develop means for measuring the mutual information.*
- *Although the approach here will focus on information-theoretic techniques, it will not be solely limited to it.*

Mutual Information

- The **mutual information** measures the amount of information that one data set conveys about another data set.
- By labeling data samples as **states** or **symbols**, one can apply mutual information measures to determine the amount of information transmitted by a channel.
- Mutual information measures can also be used to . . .
 - compare different sampling procedures
 - determine whether a given sampling procedure is sensitive to particular features in the data being sampled

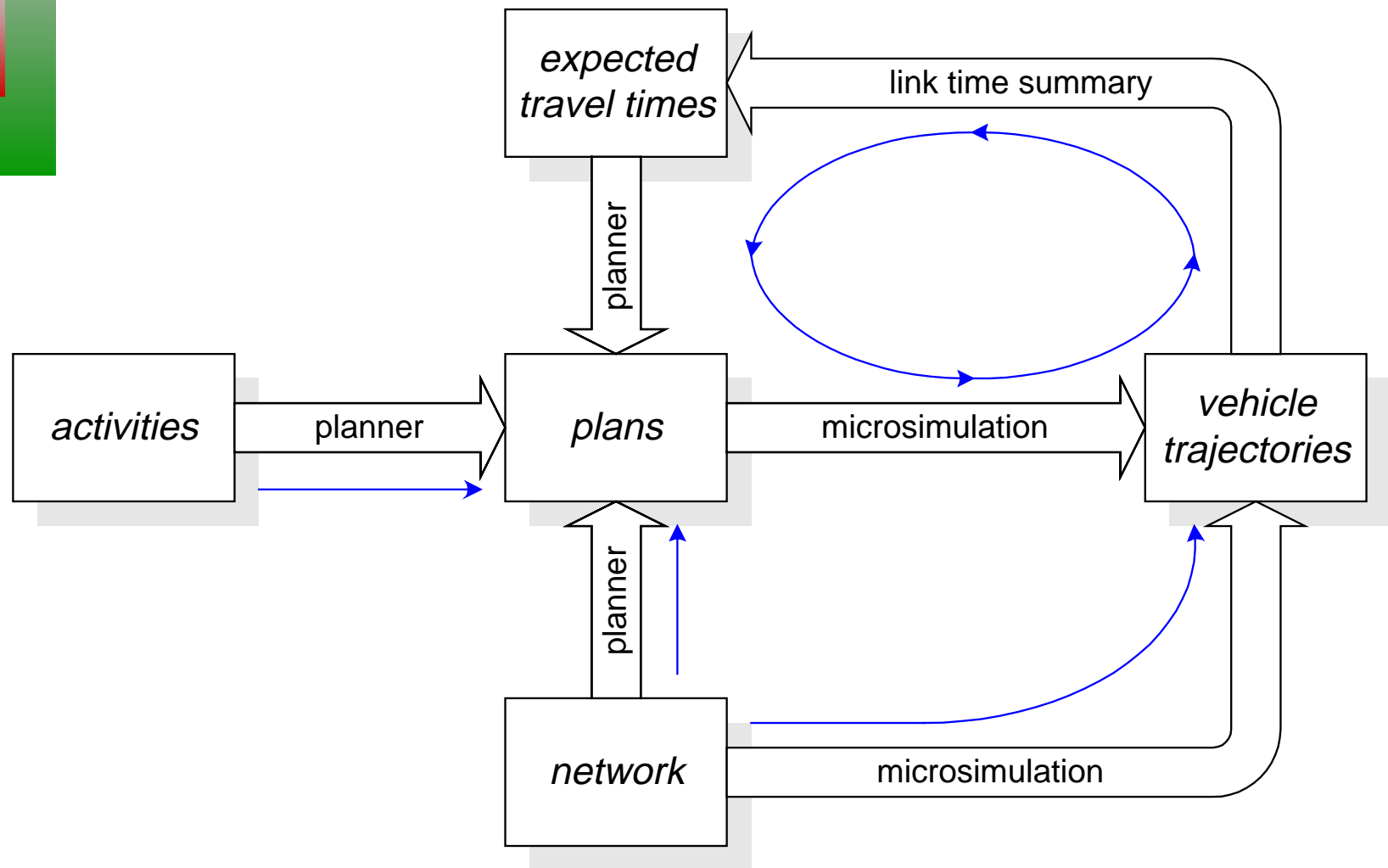




Information Flow in TRANSIMS

- *Various components of simulation frameworks (such as the planner and microsimulation in TRANSIMS) transmit information via input and output data.*
- *The feed-forward or feed-back between TRANSIMS components can be considered as the sampling of one component's output for use as the input of another component.*
- *Methods for evaluating sampling procedures can also be applied to the question of information flow through simulations.*
- *An understanding of the flow of information through a simulation framework can provide useful knowledge concerning processes within simulations such as TRANSIMS.*

Information Flow in TRANSIMS





TRANSIMS Planner Issues

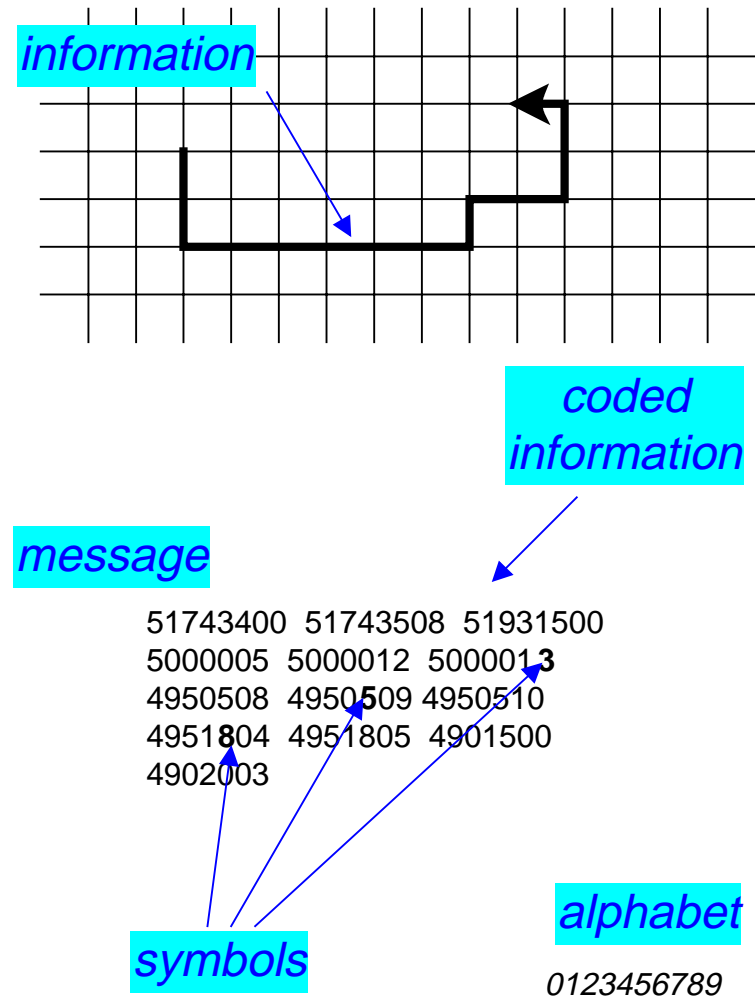
- *The problem of output sampling/compression is perhaps more important for the planner than for the microsimulation.*
 - *The microsimulation currently requires complete and detailed plans for every traveler as input from the planner.*
 - *The planner currently requires only aggregated travel times as optional input from the microsimulation.*
- *Feature extraction techniques can indicate what information is actually transmitted from the planner to the microsimulation.*
 - *How much repetition of behavior there is in the current plan sets?*
 - *It may be possible to generate generic plans shared by numerous vehicles, partially incomplete plans, or “packet-routed” plans that correctly reproduce plan set features (e.g., origin-destination tables, intersection turn counts, vehicle-miles, and vehicle-hours) in the microsimulation.*



Application: Compressed Storage of Plans

- *Plan data currently requires a lot of storage, whether on disk or in memory.*
 - *It takes 100-250 MB (depending on file format) to store the ~300,000 plans used in the DFW case study; these represent the plans that start between 5AM and 10AM and pass through the 25 square mile study area.*
 - *There are 5,886,500 person-trips per day in the Portland area.*
- *Information-theoretic techniques can be applied to plan sets to understand the redundancy in and other characteristics of plans.*
- *We find that practical methods of plan compression can significantly reduce the amount of storage (in memory or on disk) needed to store plans without negatively impacting other aspects of performance.*

-



Plan Storage Methods: Link ID (ASCII)

- *This method simply stores the succession of link IDs (numbers between 1 and $2^{32} - 1$) as space-delimited ASCII text.*
- *This is similar to the TRANSIMS ASCII planner output file format.*
- *Information is stored inefficiently because many of the symbols in the alphabet (the ASCII character set) never appear in the message.*
- *Example:*
51743400 51743508 51931500 5000005 5000012 5000013
4950508 4950509 4950510 4951804 4951805 4901500 4902003
(848 bits)

Plan Storage Methods: Link ID (binary)

- *This method stores the succession of link IDs as a series of 32-bit binary words.*
- *This is similar to the current TRANSIMS binary plan file format.*
- *Information is stored inefficiently because many of the symbols in the alphabet (32-bit binary words) never appear in the message.*
- *Example:*
 $03158AA8_{16}$ $03158B14_{16}$ $0318696C_{16}$ $004C4B45_{16}$ $004C4B4C_{16}$
 $004C4B4D_{16}$ $004B89EC_{16}$ $004B89ED_{16}$ $004B89EE_{16}$ $004B8EFC_{16}$
 $004B8EFD_{16}$ $004ACA7C_{16}$ $004ACC73_{16}$
(416 bits)

Plan Storage Methods: Unique Link ID (binary)

- A typical network only uses a fraction of the possible link IDs (3 parts-per-million in the plan set we looked at).
- One can reduce the storage required for link IDs by temporarily renumbering them consecutively when they are stored.
- Information is stored inefficiently because strong correlations exist between consecutive alphabetic symbols (links) in the message.
- Example:
 $013A_{16} \ 9B39_{16} \ 7268_{16} \ 0021_{16} \ DD3F_{16} \ 42F7_{16} \ 8A11_{16} \ 92FA_{16}$
 $73E9_{16} \ F2F0_{16} \ 5529_{16} \ 6480_{16} \ 4AA1_{16}$
where unique link IDs map into original link IDs via
 $013A_{16} \rightarrow 03158AA8_{16}$, $9B39_{16} \rightarrow 03158B14_{16}$, etc.
(208 bits)

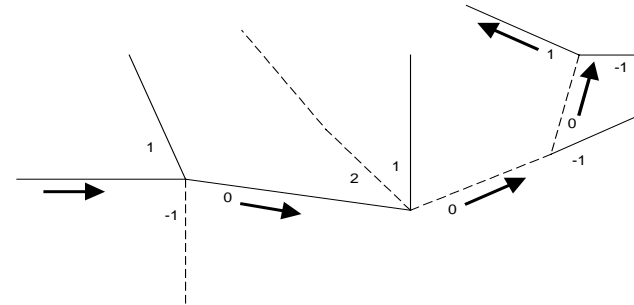
Plan Storage Methods: Run-Length Link Encoding (binary)

- *Vehicles mostly travel straight through intersections: about 80% of the movements in the plan set we looked at are straight-through movements.*
- *One can reduce the number of links stored by assuming the straight-through movement as the default at an intersection and storing only non-default movements.*
- *Information is stored inefficiently because correlations exist between consecutive alphabetic symbols (links) in the message.*
- *Example:*

*03158AA8₁₆ [03158B14₁₆ 0318696C₁₆] 004C4B45₁₆ [004C4B4C₁₆
004C4B4D₁₆ 004B89EC₁₆ 004B89ED₁₆ 004B89EE₁₆] 004B8EFC₁₆
004B8EFD₁₆ [004ACA7C₁₆] 004ACC73₁₆
(160 bits)*

Plan Storage Methods: Turn (binary)

- Turns are numbered relative to the straight-through direction, with consecutive positive integers assigned to left turns and with consecutive negative integers assigned to right turns.
- This method stores the turns in 4-bit binary half-bytes.
- Information is stored inefficiently because the frequency of the symbols is highly non-uniform.
- Example:
 $0000_2\ 0000_2\ 0001_2\ 0000_2\ 0000_2\ 0000_2\ 0000_2\ 0000_2\ 0010_2\ 1001_2$
 $0000_2\ 0001_2$
where turn 0000_2 coming from link 51743400 means to go to link 51743508, turn 0001_2 coming from link 51931500 means to go to link 5000005, etc.
(48 bits)



Plan Storage Methods: Huffman Turn Compression (binary)

- *Certain series of turns in a plan set are more common than other series.*
- *One can construct a binary representation of the series where the number of bits needed to represent a series of turns is inversely related to the frequency of the series.*
 - *Measure the frequency of all of the turn series of a given length.*
 - *Each series is considered a symbol in the alphabet from which the plans are constructed.*
 - *The optimal binary encoding of the alphabet for the plan set is the **Huffman Code** for the alphabet.*
- *The alphabetic symbols in the message are at terminal nodes of a binary tree.*
 - *The depth of a symbol is related to its relative frequency by $d_i \sim -\log_2 f_i$.*
 - *The overall number of bits necessary to encode the message is $\sum_i f_i d_i \sim -\sum_i f_i \log_2 f_i = H$, i.e., the message's entropy.*

Plan Storage Methods: Huffman Turn Compression (binary)

- Consider single turns (the length of the turn series is unity).
- Measure the frequencies of the symbols.
- Construct the Huffman code by combining the least-frequent pairs iteratively into a binary tree:

turn -3 $\rightarrow 010010_2$

turn -2 $\rightarrow 01000_2$

turn -1 $\rightarrow 00_2$

turn 0 $\rightarrow 1_2$

turn +1 $\rightarrow 011_2$

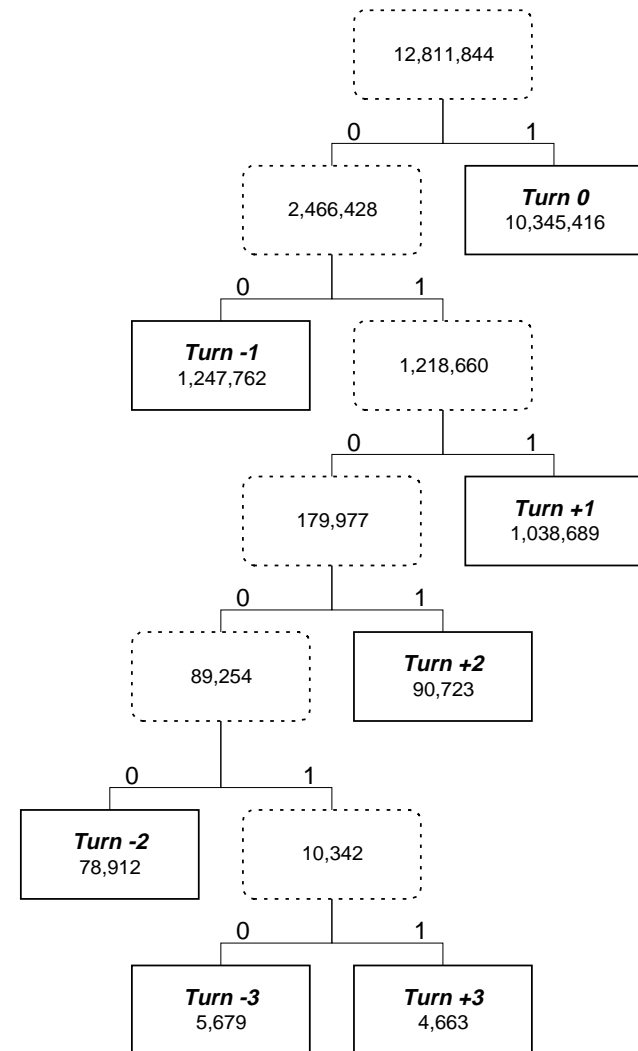
turn +2 $\rightarrow 0101_2$

turn +3 $\rightarrow 010011_2$

- Example:

$1_2 1_2 011_2 1_2 1_2 1_2 1_2 1_2 0101_2 00_2$

$1_2 011_2$ (20 bits)



Comparison of Plan Storage Methods

<i>Storage Method</i>	<i>Advantages</i>	<i>Disadvantages</i>
Link ID (ASCII)	<ul style="list-style-type: none">• viewable with text editor• link queries possible	<ul style="list-style-type: none">• requires a lot of storage
Link ID (binary)	<ul style="list-style-type: none">• link queries possible	<ul style="list-style-type: none">• requires a lot of storage
Unique Link ID (binary)	<ul style="list-style-type: none">• link queries possible	<ul style="list-style-type: none">• requires moderate storage
Run-Length Encoding of Link ID (ASCII)	<ul style="list-style-type: none">• eliminates simple redundancy	<ul style="list-style-type: none">• requires moderate storage• link queries not possible• hard to detect invalid plans
Turn (ASCII)	<ul style="list-style-type: none">• viewable with text editor	<ul style="list-style-type: none">• link queries not possible
Turn (binary)	<ul style="list-style-type: none">• requires little storage	<ul style="list-style-type: none">• link queries not possible
Huffman Turn Compression (binary)	<ul style="list-style-type: none">• requires near-minimal storage• impossible to code invalid plans	<ul style="list-style-type: none">• link queries not possible• preprocessing required before storage



Plan Set for Comparison Study

- *file: /transims/output2/segovia/PlanSets/12-25-96-k5*
- *247,620,952 bytes (TRANSIMS ASCII format)*
- *98,422,112 bytes (TRANSIMS binary format)*
- *294,702 plans*
- *13,112,498 steps*
- *12,817,796 turns*
- *14,750 links*

Plan Storage Comparison Study Results

<i>Storage Method</i>			<i>Estimated Plan Set Size (MB)</i>
Link ID (ASCII)			91.79
Link ID (binary)			52.45
Unique Link ID (binary)			22.70
Run-Length Encoding of Link ID (ASCII)			17.67
Turn (ASCII)			12.81
Turn (binary)			4.81
Huffman Turn Compression (binary)	<i>Series Length</i>	<i>Number of Unique Series</i>	
	1	7	1.56
	2	54	1.62
	3	268	1.61
	4	949	1.58
	5	2778	1.56
	6	6945	1.53
	7	14924	1.50
	8	28789	1.47



Ongoing Work in Plan Storage

- *Considering additional storage methods, including “lossy” methods that discard information.*
 - *Take better account of the correlation in turn series by allowing variable-length turn series; this will reduce the number of low-frequency series in the Huffman code.*
 - *Create a separate length-one or length-two Huffman code for each intersection—so far we have only considered codes globally applicable to the whole road network.*
 - *Construct a Huffman code based on link ID.*
- *Studying how to store efficiently expected link arrival times in plans.*
- *Packaging the Huffman plan storage and compression algorithm for use in the TRANSIMS production code.*

TRANSIMS Microsimulation Issues

- *How does one extract traffic features from simulation output?*
 - *There are numerous ways to configure vehicle sensors on a road network, for example, but only some of these are capable or efficient at detecting traffic features such as traffic jams, stopped vehicles, phase failures at traffic signals, etc.*
- *Where is information being generated in the microsimulation?*
 - *What is the relative importance of vehicle interaction at intersections versus along the roadways, for example.*
 - *How do the plans, signal timing, driving logic, the intersection model, and the roadway model each play a role in the generation and transmission of information in the simulation?*
- *What is the most efficient way to compress raw output data for storage and retrieval?*
 - *We need to reduce the disk storage required for simulation output and to speed up the parts of the simulation involving output collection without losing the traffic features present in the output.*



Caveat

- *Note that in the previous discussions, we consider the question of sampling the microsimulation—not sampling traffic.*
- *The process of microsimulation can introduce spurious information not present in the “real” traffic into the simulation output.*
- *We do not propose to study such **aliasing** issues yet.*



Application: Microsimulation Information Content

- *Microsimulation trajectory/evolution data currently requires immense amounts of storage.*
- *Information-theoretic techniques can be applied to microsimulation output to . . .*
 - *understand its information content and*
 - *locate patterns in it.*
- *A block configuration formalism provides a means to . . .*
 - *look for patterns in the CA*
 - *measure information content*
 - *perform data compression*

Labeling States

- The velocity in the **cell** (x,t) is written $v_{x,t}$ and is subject to the constraint $0 \leq v_{x,t} \leq v_{max}$.

- Each cell in the CA has $k = v_{max} + 2$ possible values.

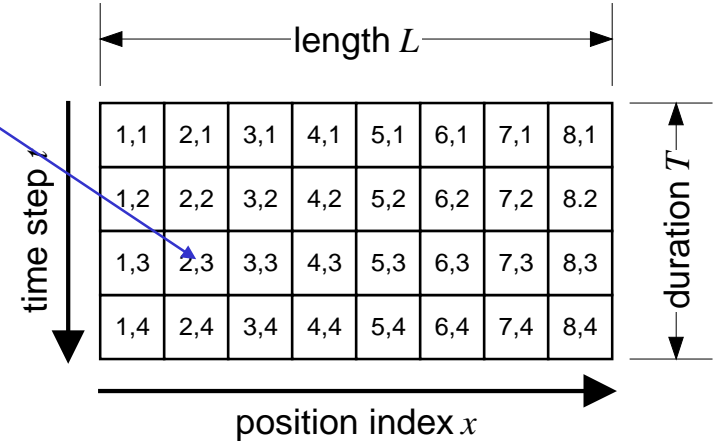
- We label each single-cell configuration by an integer:

$$c_{x,t} = \begin{cases} v_{x,t} + 1 & \text{if there is a vehicle at } (x,t) \\ 0 & \text{otherwise} \end{cases}$$

- We also label a **block configuration** $\{c\}$ of length L and duration T by an integer:

$$j(\{c\}) = 1 + \sum_{t=1}^T \sum_{x=1}^L c_{x,t} k^{(t-1)L+x-1}$$

- This scheme of labeling states does not take account of the distinguishability of vehicles.



```

0 6 0 0 3 0 0 0 4 0 0 0 0 0 0 0 0 6 0 0 3 0 0 0 4 0 0 0 0 0 0
0 0 0 3 0 0 0 4 0 0 0 0 5 0 0 0 0 0 3 0 0 0 4 0 0 0 0 5 0 0
5 0 0 0 0 4 0 0 0 4 0 0 0 0 5 0 0 0 0 4 0 0 0 4 0 0 0 4 0 0 0
0 0 0 0 0 6 0 0 0 4 0 0 0 4 0 0 0 0 0 6 0 0 0 4 0 0 0 4 0
0 5 0 0 0 0 0 4 0 0 3 0 0 0 0 5 0 0 0 0 0 4 0 0 3 0 0 0 0
0 0 0 0 0 5 0 0 0 2 0 0 0 3 0 0 0 0 0 5 0 0 0 2 0 0 0 3 0
4 0 0 0 0 0 3 0 0 0 3 0 0 0 4 0 0 0 0 0 3 0 0 0 3 0 0 0 4
0 0 4 0 0 0 0 0 0 4 0 0 0 4 0 0 0 0 4 0 0 0 0 4 0 0 0 4
0 0 5 0 0 0 0 5 0 0 0 0 4 0 0 0 0 5 0 0 0 0 5 0 0 0 0 4
0 5 0 0 0 0 5 0 0 0 0 6 0 0 0 0 5 0 0 0 0 5 0 0 0 0 6 0
5 0 0 0 4 0 0 0 0 0 6 0 0 0 0 5 0 0 0 4 0 0 0 0 6 0 0 0
0 0 4 0 0 0 4 0 0 0 0 0 0 5 0 0 0 4 0 0 0 4 0 0 0 0 0
0 3 0 0 0 3 0 0 0 0 5 0 0 0 0 3 0 0 0 3 0 0 0 0 5 0 0
6 0 0 0 4 0 0 3 0 0 0 0 0 0 6 0 0 0 4 0 0 3 0 0 0 0 0
0 0 4 0 0 3 0 0 3 0 0 0 0 0 0 4 0 0 3 0 0 3 0 0 0 0
0 0 0 0 3 0 0 3 0 0 3 0 0 0 0 0 0 3 0 0 3 0 0 3 0 0
0 0 0 0 0 2 0 0 0 3 0 0 4 0 0 0 0 0 2 0 0 0 3 0 0 4
0 4 0 0 0 0 0 3 0 0 0 4 0 0 0 4 0 0 0 0 3 0 0 0 4
4 0 0 0 0 5 0 0 0 0 4 0 0 0 4 0 0 0 5 0 0 0 0 4 0 0
0 0 0 5 0 0 0 0 6 0 0 0 5 0 0 0 5 0 0 0 0 6 0 0 0
0 0 5 0 0 0 0 6 0 0 0 5 0 0 0 5 0 0 0 0 6 0 0 5
0 0 5 0 0 0 5 0 0 0 0 5 0 0 0 5 0 0 0 5 0 0 0 5
4 0 0 0 0 0 5 0 0 0 0 6 0 0 4 0 0 0 0 5 0 0 0 0 6 0
    
```

Escort Distribution

- We use the method of **escort distributions** to probe the structure of this probability distribution.
- Let p_j be the probability of observing the state j in a sample of length L and duration T .
- The **escort probabilities** associated with the p_j are:

$$P_j = \frac{(p_j)^\beta}{\sum_{i=1}^k (p_i)^\beta} = \exp[\beta(F_\beta - b_j)]$$

- $\beta = 1$: the escort probability is identical to the probability
 - $\beta = 0$: all states have equal probability
 - $\beta \rightarrow +\infty$: the largest probability dominates
 - $\beta \rightarrow -\infty$: the smallest probability dominates
- We define the **bit number** for the state j as:
 $b_j = -\ln p_j$

Rényi Entropy

- **Rényi entropy** (“free information”)

$$S_{\beta}(X, T) = \frac{\beta}{1-\beta} F_{\beta}(X, T) = \frac{1}{1-\beta} \ln \sum_{j=1}^{k^{XT}} (p_j)^{\beta}$$

- **special cases**

- **set entropy**

$$S_0(X, T) = \lim_{\beta \rightarrow 0} S_{\beta}(X, T) = \ln \sum_{j=1}^{k^{XT}} \theta(p_j) \quad \text{where} \quad \theta(x) = \begin{cases} 1 & \text{for } x > 0 \\ 0 & \text{otherwise} \end{cases}$$

- **measure entropy** (or Shannon entropy)

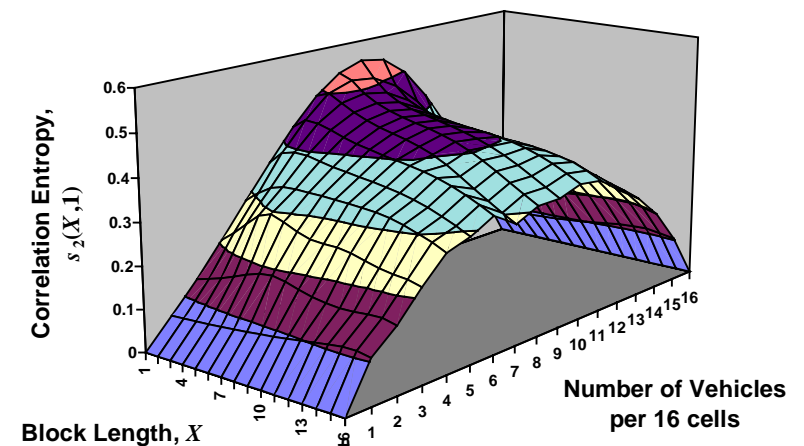
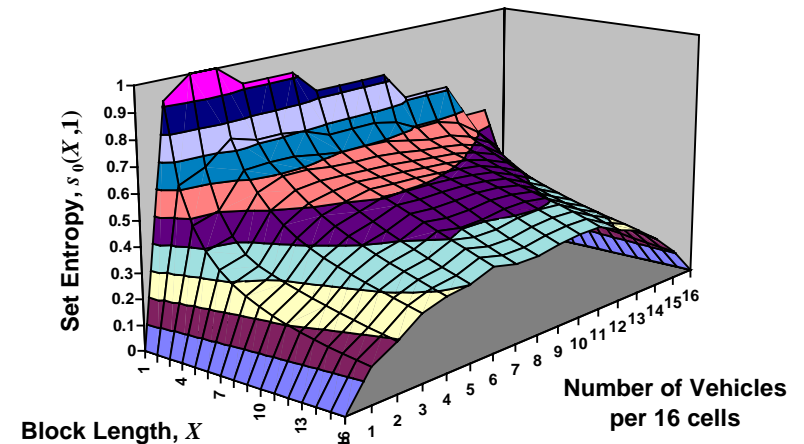
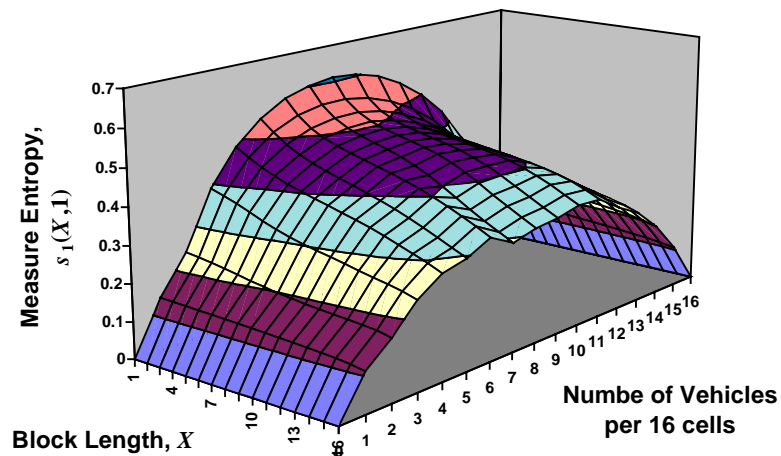
$$S_1(X, T) = \lim_{\beta \rightarrow 1} S_{\beta}(X, T) = - \sum_{j=1}^{k^{XT}} p_j \ln p_j$$

- **correlation entropy**

$$S_2(X, T) = - \ln \sum_{j=1}^{k^{XT}} (p_j)^2$$

Numerical Experiments

- We have collected a data set of 80,000 samples from a simple one-lane CA operating under the TRANSIMS driving rules.
- The measure entropy provides an indication of how much data compression is possible when storing microsimulation trajectory/evolution data.



An abstract graphic consisting of several overlapping rectangular blocks. A large green block is at the bottom. Above it, a blue block is on the left, and a red block is on the right. A thin horizontal red line spans the width of the composition, positioned above the red block.

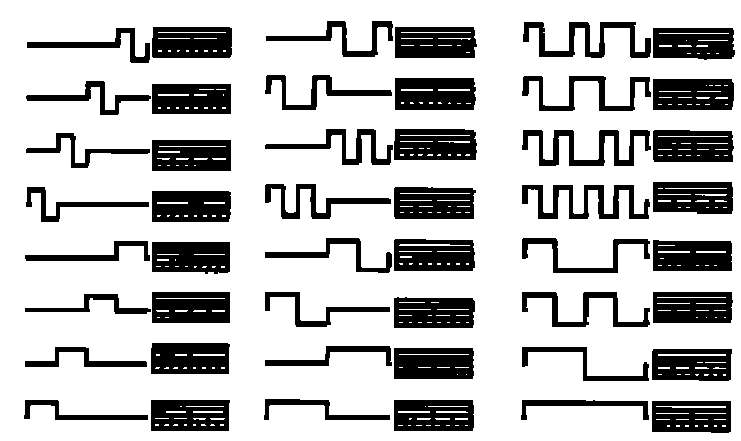




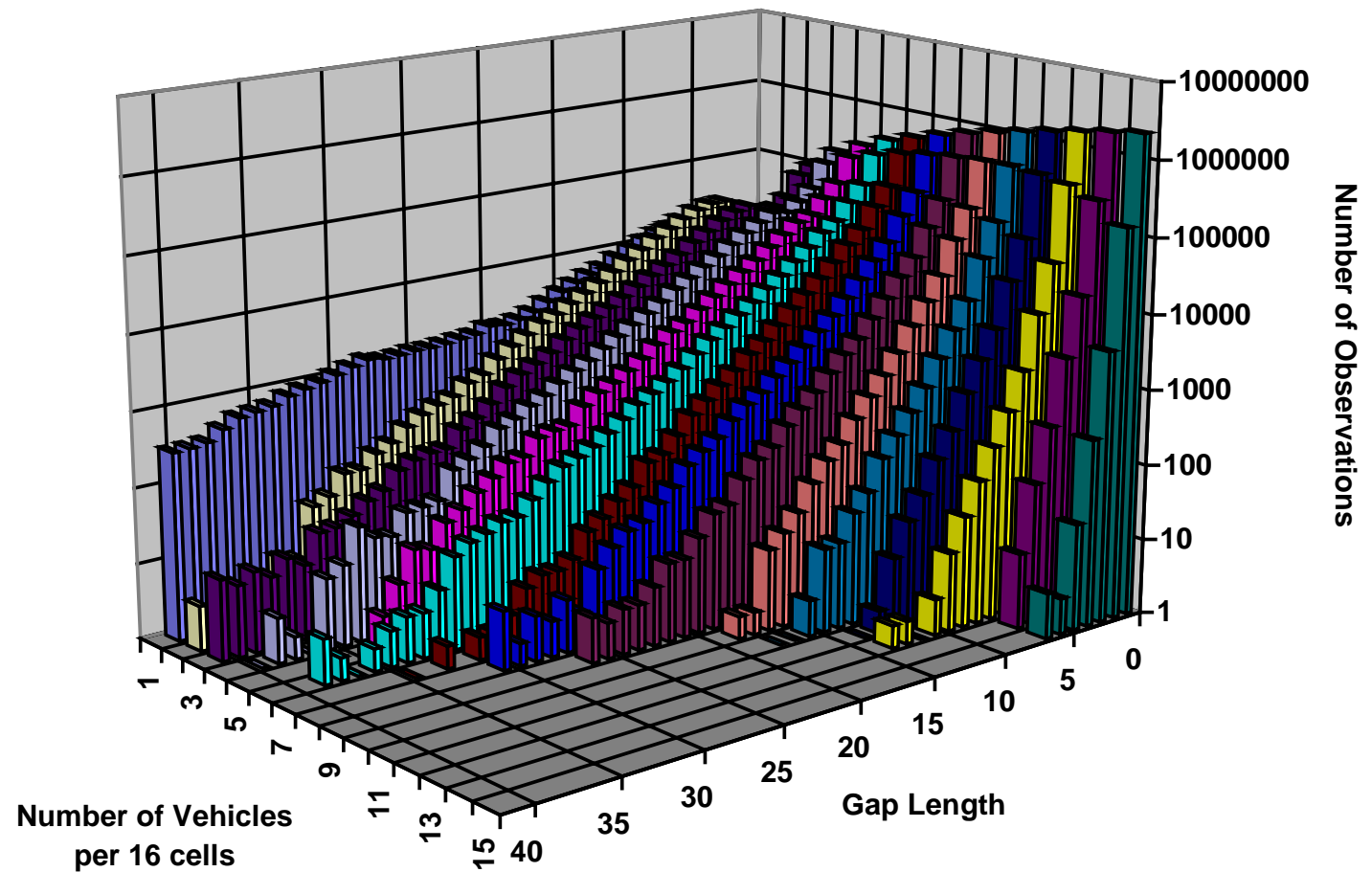
Application: Microsimulation Scaling Behavior

- *It is difficult and time-consuming to locate patterns in the microsimulation trajectory/evolution data.*
- *Multiresolution analysis provides a means to . . .*
 - *identify scaling behavior and special types of patterns in the CA*
 - *perform “lossy” data compression*

- 1



Scaling of Vehicle Gap Length





Scaling of Grid Cells

- *Instead of labeling the complete state of a group of cells, one can assign a label that aggregates the information in the group.*
 - *For instance, groups of cells can be labeled by their number of vehicles and total velocity.*
 - *This is similar to the current link box data collection in the CA.*
- *These group functions do not necessarily have to preserve vehicle counts or velocities, but it is preferable that they do not bias averages.*
 - *For example, each pair of cells can be combined into a new cell that has either zero or two vehicles an integer velocity—the states of such cells can be labeled just as single cells are.*
 - *It would be interesting to understand how the CA output behaves under such as factor-of-two scaling.*



Ongoing Work in Microsimulation Sampling

- *Considering additional storage methods, including “lossy” methods that discard information.*
 - *Multiresolution analysis using wavelets makes it possible to separate the CA behavior at a variety of space- and time-scales.*
 - *Try to retain vehicle identity during compression.*
- *Developing techniques to recognize patterns in the microsimulation, both off-line and on-line (as it is running).*
- *Measuring the link travel time information that the microsimulation adds with respect to the planner.*
- *Packaging the block symbol storage and compression algorithm for use in the TRANSIMS production code.*



Status

- *Data compression implementation efforts are underway.*
 - *“Huffman turn” compression of planner output*
 - *“block symbol” compression of microsimulation output*
- *Research continues.*
 - *information content in planner and microsimulation output*
 - *flow of information between TRANSIMS modules*
 - *identification of patterns/features in microsimulation output*



Conclusion

- *There are many ways to represent a given piece of information.*
- *Some representations convey more meaning than others*
- *We can use these facts to our advantage in TRANSIMS.*
- *New types of data sampling and storage are possible, practical, and interesting.*